Introduction to VMware ThinApp

VMware ThinApp 4.0

vmware

Introduction to VMware ThinApp
Revision: 20080627
Item: EN-000063-00

You can find the most up-to-date technical documentation on our Web site at:

http://www.vmware.com/support/

The VMware Web site also provides the latest product updates.

If you have comments about this documentation, submit your feedback to:

docfeedback@vmware.com

VMware, Inc.
3401 Hillview Ave.
Palo Alto, CA 94304
www.vmware.com

Contents

Introduction to VMware ThinApp

VMware ThinApp enables you to package, run, and manage your software applications. ThinApp captures the installation of applications into a single executable file and enables you to install your captured applications with no dependencies on the host personal computer (PC). ThinApp does this by *virtualizing* the application.

Application virtualization enables the deployment of software without modifying the local operating system or file system. It allows you to deliver and update software in an isolated environment while ensuring the integrity of the operating system and all applications. This significantly reduces application conflicts and the need for regression testing. A single application can be bundled and deployed to multiple operating system versions. Applications are easier to provision, deploy, upgrade, and roll back.

A comparison with machine virtualization can be useful to help understand application virtualization. Machine virtualization, as accomplished with VMware products like Workstation and ESX, decouples operating systems from the underlying hardware. This is done by abstracting the physical hardware resources and presenting them to the operating system as virtual resources. The result is a higher degree of hardware independence, isolation, and encapsulation than can be achieved with operating systems installed directly onto hardware resources.

Application virtualization with VMware ThinApp decouples applications from the underlying operating system. Abstracted operating system resources are presented to the application as virtual resources. The result is a higher degree of independence, isolation, and encapsulation than for applications installed directly into an operating system environment. Application virtualization is highly complementary to machine virtualization. Using both machine and application virtualization together increases the benefits of each.

This document provides an introduction to ThinApp and includes the following topics:

For specific information and instructions for using ThinApp, see the *VMware ThinApp User's Manual*.

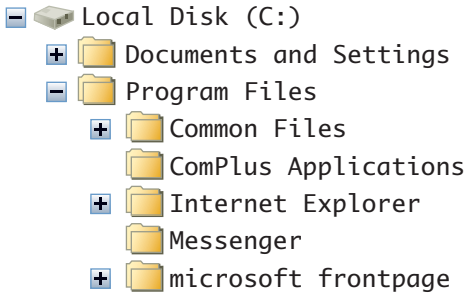
How ThinApp Works

ThinApp works by using a build process to "link" the virtual operating system with a compressed embedded file system and registry into a single executable file. The executable file can run with zero installation, and without decompressing files to disk, from any data source including your desktop, a network path, or removable storage like USB Flash and CD ROM. ThinApp enables applications to run directly from storage devices such as USB flash or network shares in an efficient manner by using block-based streaming with transparent decompression.

ThinApp accomplishes "zero installation" by presenting a virtual environment to the running application, making it appear as if all of its files, registry entries, environment variables, COM/ActiveX controls, services, and so on are already installed on the PC, even though no changes have actually been made.

The virtual environment presented to the application is a merged view of files installed by the application and files already existing on the PC. For example, consider a host PC that has a file system that looks like Figure 2-1 in Windows Explorer:

Figure 2-1. Host PC File System Viewed in Windows Explorer

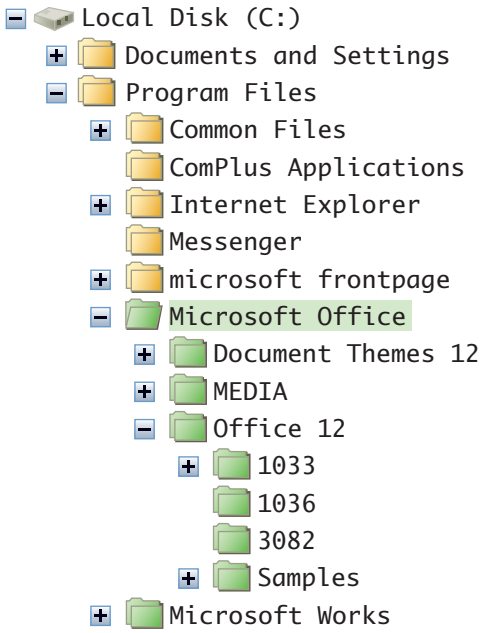


Microsoft Office creates various directories during its installation process, including these:

```
C:\Program files\Microsoft Office\...
C:\Program files\Microsoft Works\...
```

When you run a version of Microsoft Office captured by ThinApp, the application sees all of the original files on the PC plus the additional directories installed by Microsoft Office. If you select **`File > Open`**, you see the directory structure as shown in Figure 2-2:

Figure 2-2. ThinApp-Captured Microsoft Office File System Viewed in Windows Explorer



Because these directories are not actually created on the host PC, the PC remains unchanged and captured applications do not negatively impact other applications on the same PC.

ThinApp presents a merged view of the system registry as well. Figure 2-3 shows the registry as seen by Windows Regedit:

Figure 2-3. Registry as Seen by Windows Regedit

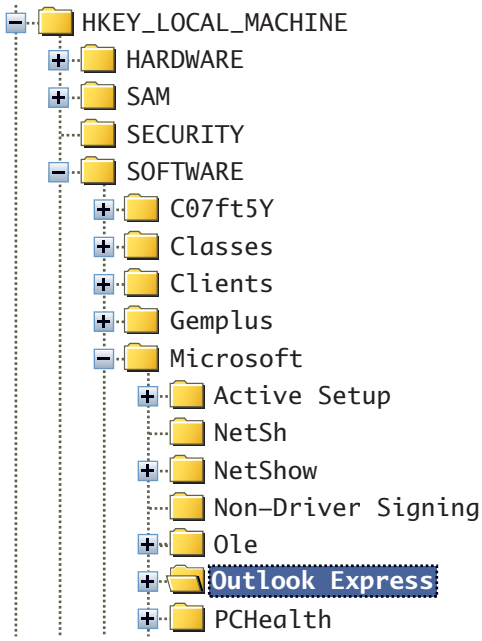
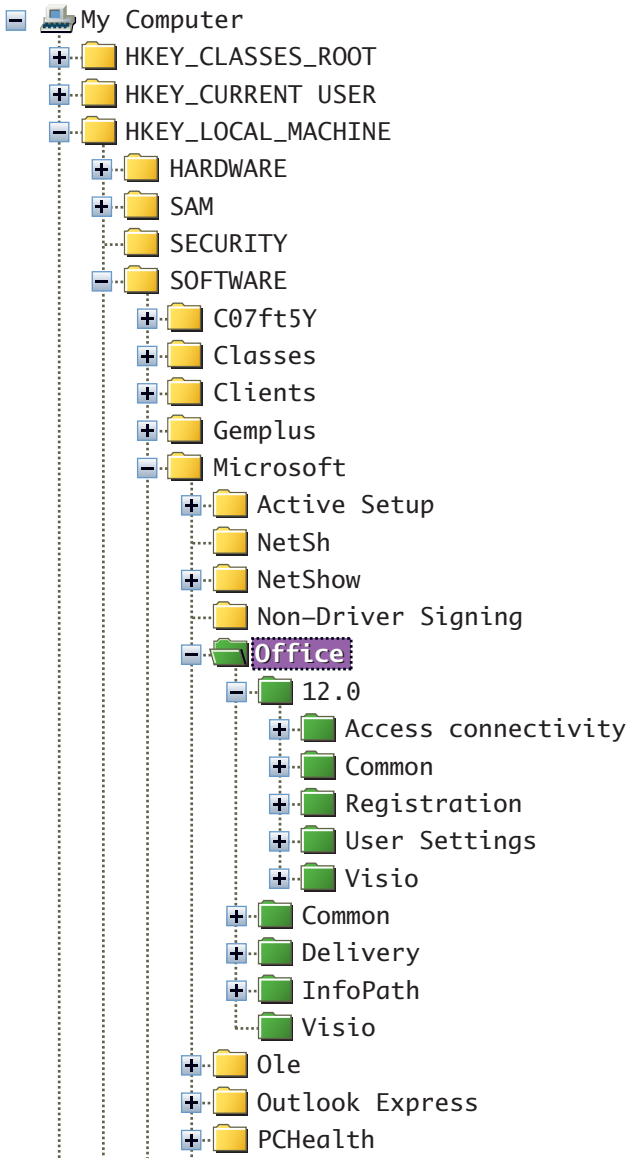


Figure 2-4 shows the registry as seen by a captured version of Microsoft Office:

Figure 2-4. Registry as seen by a captured version of Microsoft Office



Features of Thinapp

The following sections describe ThinApp features.

User Mode Operation

ThinApp runs completely in user mode, which provides many system stability, security, infrastructure, and ease-of-use advantages. For more information, see "User Mode" on page 13.

Virtual Side-By-Side (SxS) Support

ThinApp supports Side-by-side (SxS). SxS is an operating system feature supported by Windows XP, Windows 2003, and Windows Vista. SxS is required to deploy most current software products including Microsoft Office 2007, Adobe Reader 8, and .NET 2.0/3.0. For more information, see "Side-By-Side (SxS)" on page 14.

Application Isolation

ThinApp enables applications to run without any modification to the registry or file system of the host PC. Other applications running on the same PC are unaware of virtualized applications.

Multiple Simultaneous Client Application Versions

ThinApp is the only virtualization technology that supports multiple concurrent running copies of its client application on the same PC. That is, you can install Application B by using the most current release of ThinApp without affecting Application A that was installed using a previous version of ThinApp.

Each application that is captured and installed with ThinApp continues to operate independently of ThinApp itself.

Instant Portable Deployment of Applications on USB Flash or CD-ROM)

ThinApp can easily convert standard applications into portable applications that run from USB flash devices or CD-ROM players. For USB deployment, ThinApp uses its portable mode to redirect application registry and file system changes intended for the host PC to files stored on the portable device. Because ThinApp has no device drivers and runs in guest as well as restricted user accounts, you can use captured portable applications on kiosk PCs even if they are locked-down and do not permit any installation.

ThinApp Applications Are Loaded as Normal Applications

Windows views executables generated by ThinApp as normal applications that run without external dependencies. Other applications on the system are not affected by installing ThinApp virtual applications.

Setup Capture

The ThinApp Setup Capture wizard provides a simple way to package applications into virtual applications. Use Setup Capture to take snapshots of a machine before and after you install an application. The ThinApp package is created using the differences between the two snapshots.

Text-Based Settings Files

Except for application files, which are stored as normal binary files, all ThinApp files and settings are stored as .ini files so ThinApp projects can be managed using source-control applications.

Compression

ThinApp uses block-based streaming decompression. Compressed data does not need to be decompressed to a disk before it can be accessed. This means that you can launch packages from a network share without decompression. All the package data is decompressed one block at a time as needed by the application. Only startup data is sent over the network.

When you deploy packages to PC hard drives for offline use, requirements for disk space are reduced because package data remains compressed at all times. ThinApp has compression ratios similar to those achieved with ZIP compression.

Application Sync

Use Application Sync to deploy ThinApp application updates. This enables ThinApp to automatically check for and install updates to your packaged application. Updates might include changes such as a new version, service pack updates, or configuration changes in the package.ini file.

NOTE If you use Application Sync, VMware recommends that you disable automatic application updates that are configured in your virtual application. Conflicts might occur between the linked packages and the software that is automatically updated.

Application Link

Application Link connects deployed applications. For example, you can establish a relationship between a deployed instance of Microsoft Office 2003 and a new Microsoft Office plug-in. Application Link enables you to establish a link between applications without having to encapsulate them into the same executable package.

Terminal Server and MetaFrame Support

You can use the same ThinApp packages on terminal servers and desktop PCs without changes. You can run multiple versions of an application simultaneously on the same server without conflicts because of the registry and file-system isolation that ThinApp provides. You can update applications without having to stop currently running copies.

Virtual File System

ThinApp supports packages greater than 4GB in size. It also supports sandboxing, isolation, and system merge modes for specific registry subtrees. Sandboxed files can be scanned and blocked by anti-virus software. The virtual file system is stored in macro format, remapping shell folders automatically. The virtual file system automatically migrates Side by Side (SxS) from Windows XP to Windows Vista.

Virtual Registry

The ThinApp virtual registry is a visible application, not the entire system. The following virtual registry features are supported:

- Executable files contain a read-only image of the registry.
- The virtual registry is merged onto the system registry.
- Application specific registry sub trees are automatically isolated from the host PC to prevent conflicts with locally installed versions.
- The sandbox stores runtime modifications as “diffs.”
- The registry can contain per-user data (`HKEY_CURRENT_USER`).
- The registry can contain SID-specific user data.
- The registry can be reset to a captured state by deleting the sandbox.
- The virtual registry is shared by all applications in the same sandbox.
- Automatic backup and restoration occur upon disk corruption (common USB flash).
- Dynamic Path Relocation is applied to all registry reads and writes allowing instant operating system and PC migration. For more information, see “Dynamic Path Relocation” on page 15.

Virtual COM

Virtual COM provides the following features:

- In-process COM from virtual DLLS.
- No system registration.
- No extracting DLLs or OCXs to file system.
- Out-of-process COM/DCOM.
- Executable-based COM without installation.
- Service-based COM with virtual services.

Dynamic Remapping

ThinApp provides the following dynamic remapping features:

- Registry and file-system dynamically re-adjust for instant migration across operating systems ("Dynamic Path Relocation" on page 15).
- User profile data and sandbox dynamically remaps, enabling application and setting migrations.
- Execution on USB flash from PC to PC.
- Manages shell folders and short path names.

Virtual Services

ThinApp provides the following virtual-services-related features:

- Applications that require a service can be packaged.
- Virtual services auto-start when an application is launched.
- Virtual services can be started and stopped by an application like real services.
- Support for packaging services and deployment as real Windows services is started at boot time.

Scripting

ThinApp provides the following scripting features:

- Embed and enable `.vbs` in executable files.
- Configure pre-launch and shutdown.
- Expose the VMware ThinApp runtime API.
- Provide access to any COM scripting provider.
- Execute commands in a virtual or real environment.
- Set application time-outs, and authenticate with LDAP or a database.

Active Directory Authentication

ThinApp supports the following Active Directory Authentication features:

- Application access tied to Active Directory groups.
- Dynamic addition and removal of users to Active Directory groups for access control purposes.
- Offline cached credentials for offline usage.

Packaging Runtimes

ThinApp enables you to package runtimes with your application and eliminate pre-installation requirements. Some examples include:

- .NET 1.1, 2.0, 3.0
- Java
- Perl
- Crystal Reports
- COM and ActiveX controls

Security Features of ThinApp

ThinApp has the following security features:

- **Group policy security**– Because ThinApp has no kernel-mode code, it cannot violate machine group policy applied to user accounts.

  ThinApp has no ability to give applications elevated permissions for devices on a machine, such as the file system, registry, networking devices, or printers.

- **Runs in restricted user accounts** – Because ThinApp requires no device drivers, it can run applications in guest user accounts without previous installations of the software.

- **Allows applications requiring administrator rights to run without additional privileges** – When you make global changes using a virtual application, the ThinApp sandbox provides a user- and application-specific location to make those changes. The host system is not affected. This feature enables applications to run in secure, restricted environments like Terminal Server and Windows Vista.

Streaming Applications with ThinApp

ThinApp provides streaming capability without requiring a new server or client. ThinApp uses the standard SMB protocol to stream applications over a LAN, so any Windows file share can instantly become a streaming server. ThinApp embedded client technology enables you to click on executable files from network shares after which the client is loaded directly into memory.

ThinAppsupports the following streaming features:

- Windows Client
- Server can be any SMB share
- Block-by-block streaming
- Instant start for packages over 8GB in size
- Streams from any source media, including network shares and ISCSI, hard drives, USB Flash, and CD-ROM

Sandbox

The sandbox holds runtime modifications that applications make as they are running. The executable that you build never changes, so it can be placed in a shared folder with read-only access. The sandbox has the following features:

- Provides per-user and per-application storage of application modifications.
- Provides protection for the host machine since the virtual application is self-contained.
- Enables applications to run on Terminal Server if they are not already able to do so.
- Enables most applications to run on Vista if they are not already able to do so.
- Enables IT to maintain locked-down desktops.
- Reverts to a known application state when you delete the sandbox.

User Mode

VMware ThinApp uses user mode for application virtualization. User mode maintains system security, stability, and usability.

Windows runs all code in one of two modes, user mode and kernel mode (also referred to as Ring3 and Ring0 respectively). The two modes reflect two different security models that are enforced directly by your Intel or AMD processor. Code running in kernel mode has full machine access with no security controls. For example, your code can write to raw device ports, intercept, and filter system-wide file system activity, read and write machine-wide process and kernel memory, and access any kernel or process objects without regard for security descriptors. Kernel mode code is either a device driver or the Windows kernel itself.

User mode is the mode in which all applications run. User mode has strict security policies applied at all times. User mode code cannot do anything to a machine that directly causes it to fail or violate applied security policies. For example, user mode code cannot access files owned by other users unless file system permissions allow it. User mode code cannot make network connections unless the group security policy enables it to do so. Any 80386 (Pentium+) and higher processors enforce a specific security policy on user mode applications by prohibiting them from executing instructions that talk to device ports directly, preventing access to memory in other processes, and by preventing execution of kernel mode without first going through specific controlled entry points.

For user mode code, Windows supports many different sets of user accounts that are based on security policies. For example, administrator and guest accounts are user-based security policies. Administrators typically have nearly full permission to access any system objects they want, while guests are restricted and cannot read other users' files or write to global locations on a machine.

Both administrators and guests run applications in user mode, but switch to kernel mode when making system calls to access objects. Kernel-mode code verifies security descriptors to check if users have access to the objects they request.

User mode enables ThinApp to do the following:

- ThinApp can run applications on locked-down PCs without administrator rights. This means remote users can execute applications on kiosk and hotel PCs where they are not able to install software or device drivers.
- ThinApp can run applications directly from USB flash devices and various portable storage. Because ThinApp can be loaded without a client component, you can use a packaged application on any PC or network share.
- ThinApp operates with all system-level software and does not conflict with device drivers.

Improved Security in User Mode

Software applications typically have bugs that create potential security problems. For example, Internet Explorer can be used to force machines with Internet access to execute code using an HTML email or redirecting page views to specific Web sites.

With Windows Vista, Microsoft runs Internet Explorer from a separate user account that has limited security rights. Because of this, a compromised Internet Explorer account cannot do anything to the rest of the machine. This solution is possible because Internet Explorer runs in user mode.

Because ThinApp runs in user mode, any bug or vulnerability presents no additional risk to the rest of the system because all ThinApp code is running in user mode in the same security context as the application.

Other solutions use device drivers and run significant amounts of kernel-mode code. If these solutions are compromised, you can lose full control over your machine. User mode code can be walled off using user accounts but kernel-mode code cannot.

Because ThinApp runs in user mode, it has the same rights and permissions as any other application a specific user has. ThinApp cannot exceed the security rights of the user account it is running in because it has no device drivers or components running in kernel mode.

When using VMware ThinApp, administrators do not need to consider how their system-wide security policies are affected.

VMware ThinApp can be deployed in conjunction with central IT group policies or separately in the case of smaller development groups. Developer groups can use the code components and frameworks of their choice without requiring security policy changes by their IT organization.

Improved System Stability in User Mode

System failures can be hard to diagnose without significant time and resources. Most system failures are caused by third-party device drivers.

Every instruction running in kernel mode represents a risk of system failure. Often failures are caused by thread deadlocks that only occur under rare circumstances and when combined with other products. Kernel-mode applications have no protection from system failure or leaked resources. A kernel-mode system failure is often caused by a deadlocked thread, so that the machine appears to be frozen or performs badly for no apparent reason.

User mode code can never directly cause a system failure and can be easily stopped with a task manager if it causes the system to freeze. There is no other impact on the system. Windows automatically cleans up after a failure by closing open file handles, freeing allocated memory, and discarding created resources.

User mode enables you to run applications directly from locked down accounts without preinstalling a device driver or granting the user elevated security privileges.

Because ThinApp has no device drivers, it enables deployment of applications with zero footprint on desktop PCs. Applications can run from network shares with no preinstallation requirements.

Side-By-Side (SxS)

VMware ThinApp supports Side-by-side (SxS). SxS is an operating system feature supported by Windows XP, Windows 2003, and Windows Vista. SxS is required to deploy most current software products including Microsoft Office 2007, Adobe Reader 8, and .NET 2.0/3.0.

With SxS technology, applications can install DLLs to version-specific directories. This informs Windows which version of the DLL should be used when that DLL is loaded. For example, Microsoft Office 2007 installs MFC80 (8.0.50727.42) to the following path:

```
%SystemRoot%\WinSxS\x86_Microsoft.VC80.MFC_1fc8b3b9a1e18e3b_8.0.50727.42_x-ww_dec6ddd2\mfc80.dll
```

The path contains information about the DLL version. The only way to place files at this location is by using MSI installer technology. An installed SxS DLL is installed in different locations on Windows XP and Windows Vista. Because users might migrate their existing Windows XP installations to Vista, the upgrade process automatically moves the DLLs to their new location on Vista.

Windows operating systems that have SxS look for an application's manifest information to determine which version to load from SxS. A manifest is an XML description of all the SxS DLLs that can be loaded by an application. The manifest also states which version of those DLLs to use. The manifest can be embedded in the application executable or DLL in the file as a resource. The manifest can also be stored separately on the file system as a `.manifest` file. The manifest file for `app.exe` is called `app.exe.manifest`.

The manifest resolution algorithm provides search capabilities based on the user's language (for example, French and German versions are available). SxS DLLs have a force upgrade mechanism implemented as separate security policy files (also XML files). This mechanism enables Microsoft to override version number requests by applications if a security hole is discovered in a shared library. For example, an application manifest might request version 1.0.0.0 of a given DLL, but Microsoft can provide a policy file update that redirects all requests of version 1.0.0.0 to 1.0.0.5.

If the application normally installs DLLs to `c:\windows\winsxs\...`, this is reflected in the capture and appears in your project under `%systemroot%\winsxs\...`.

At runtime, ThinApp examines an application's manifest file (from the resource section or separate `.manifest` virtual file) and determines which version of a DLL to use. ThinApp then loads the correct version of the DLL from the virtual path (contained inside the package). VMware ThinApp supports management of runtime activation contexts as well, so it knows which version of DLLs to load from dynamic DLL loads.

Natively (without ThinApp), SxS is supported on Windows XP, Windows 2003, and Windows Vista, but not Windows 2000 and Windows NT. If you want to support Windows 2000 and Windows NT without ThinApp, you install your virtual application at the following locations (this is automatically performed by MSI installer 3.0):

- `c:\winnt\system32` (the application directory)
- `c:\winnt\winsxs` (SxS path location)

Installation of your virtual application at the first location is required to support Windows 2000 and Windows NT. Installation of the virtual application at the second location is required to meet Microsoft's "Designed for Windows" guidelines. These guidelines state that the application must continue to work if the operating system is upgraded from Windows 2000 to Windows XP.

Because VMware ThinApp has its own SxS processing, all SxS technology is available for Windows NT, Windows 2000, Windows XP, Windows 2003, and Windows Vista in the VMware ThinApp environment.

On Windows Vista, Microsoft has changed the path locations and file path name-altering algorithm for SxS DLLs. VMware ThinApp automatically moves SxS DLLs captured for Windows XP and Windows 2003, placing them in the correct location in virtual space for Windows Vista at runtime when the application starts. Therefore, the same executable can run on all platforms with no changes.

VMware ThinApp is the only virtualization technology that fully supports SxS which allows you to virtualize most new applications in which virtualization products fail because they must have SxS DLLs physically installed on the machine before they work.

Dynamic Path Relocation

Dynamic path relocation is the ability to move files and modify registry values to match the local host PC. ThinApp performs dynamic remapping during application startup and at runtime. Dynamic remapping enables both applications and their associated settings to migrate across different versions of Windows.

Instant SxS DLL Migration

Windows XP SxS DLLs are migrated to Windows Vista SxS DLLs automatically depending on the platform you are using.

If you capture an application that uses SxS DLLs on Windows NT, Windows 2000, Windows XP, or Windows 2003, it installs SxS DLLs to a path location different than when installed on Vista. ThinApp dynamically moves these SxS files during application startup if the platform has changed. Using ThinApp dynamic path relocation, you can create one package that works on all platforms.

File System Shell Folder Remapping

Many applications access files using shell folder locations. For example, applications typically call `GetWindowsDirectory` to obtain the path to `c:\windows` instead of using a hard-coded path. On different versions of Windows, the system directory is located in different locations. In addition, the user can select an alternate directory during installation of Windows. Applications also typically use `shfolder.dll` to obtain the path various shell folder locations like `c:\Program Files` and `c:\documents` and `settings\username`.

An example is Macromedia Flash, which installs to `c:\windows\system32\macromed\flash`. At runtime, Flash uses `GetWindowsDirectory` to obtain the partial path `c:\windows\system32`, and then appends `macromed\flash` to obtain the location of its installation directory. In addition, Flash uses a registry value that corresponds to the following location:

```
HKEY_CLASSES_ROOT\CLSID\{1171A62F-05D2-11D1-83FC-00A0C9089C5A}\InprocServer32 DefaultValue =
C:\WINDOWS\system32\Macromed\Flash\Flash9b.ocx
```

When an application is running that installs Macromedia Flash dynamically or during the capture process (for example, Firefox or Internet Explorer), the registry stores the path `C:\WINDOWS\system32`, and files are written to `c:\windows\system32\macromed\flash`. If the application is moved to another PC where the Windows root directory is different (for example `c:\winnt` on Windows 2000), the application fails to work unless both the files and registry keys are remapped to point to `c:\winnt`.

The ThinApp virtual file system stores file paths using folder macros, so the file paths automatically point to the correct location on different PCs. ThinApp stores registry data using the same folder macros, so that registry values automatically readjust to point to the correct location on a different PC. For example, when the application writes the registry value `C:\WINDOWS\system32\Macromed\Flash\Flash9b.ocx`, ThinApp stores this internally as `%SystemSystem%\Macromed\Flash\Flash9b.ocx`. When the application queries for this value, it transparently expands back to `C:\WINDOWS\system32\Macromed\Flash\Flash9b.ocx` when running on Windows XP, Windows 2003, Windows Vista, and `C:\winnt\system32\Macromed\Flash\Flash9b.ocx` when running on Windows 2000 or Windows NT.

Registry Data Remapping

ThinApp intercepts all data written to the registry and seeks references to short path names or shell folders. If any references are found, it internally stores the registry data in macro format so that the data re-expands to the correct location on other PCs.

Short Path Names

ThinApp sits above the Windows loader and file system, and correctly handles short path names (DOS 8.3 file names).

For example, applications like Microsoft Office have a large number of registry values that contain entries like `C:\PROGR~1\MICROS~3`, but on other computers, the virtual files might actually appear to exist at `C:\PROGRA~1\MICROS~4`. Because of this, various COM components fail to work at runtime.

Many applications do not install or run properly when a non-default path is used. ThinApp uses dynamic macro expansion for all registry and filename information to address the short path name issue.

At runtime, ThinApp filters registry and filename data to replace short paths with macro versions that re-expand to the correct location on new computers. In this scenario, registry values automatically readjust to point to the correct locations when the package is run on a different computer. Short path names are DOS 8.3 compatible names that map into their long path name equivalents. For example `C:\PROGR~1` is the short path name version of `C:\Program Files`.

Short path names are important for most applications for several reasons:

- Short path names eliminate spaces from paths that prevent some compatibility and security issues. For example, when executing the command `C:\Program Files\Microsoft Office\OFFICE11\winword.exe c:\Myfile.doc` using ShellExecute, Windows attempts a number of possible commands to see if they are valid:

  ```
  c:\Program.exe "Files\Microsoft Office\OFFICE11\winword.exe c:\Myfile.doc"
  c:\Program Files\Microsoft.exe "Office\OFFICE11\winword.exe c:\Myfile.doc"
  c:\Program Files\Microsoft.exe Office\OFFICE11\winword.exe "c:\Myfile.doc"
  ```

  If you create the file `c:\Program.exe` on a PC, it might causes problems when you are using Microsoft Office and might also create a security problem.

- Short path names enable compatibility with legacy applications—16-bit applications must use DOS 8.3 filename paths. When Windows runs 16-bit applications, it provides applications with the short path name version for all filenames.

- Short path names work around Windows API path length limitations. Many Windows API functions have limitations on the maximum string length they can handle for paths. For example, on Windows XP SP1, a ShellExecute API command cannot handle a string length longer than 128 characters. Often the registry is used to store a value such as Command + Parameters. The following entry for Office 2003 is an example:

  ```
  HKEY_LOCAL_MACHINE\SOFTWARE\Classes\Applications\ois.exe\shell\Edit\command DefaultValue =
  C:\PROGRA~1\MICROS~3\OFFICE11\OIS.EXE /shellEdit "%1"
  ```

  If the executable file name plus the parameter is greater than 128 characters, the command fails on Windows XP SP1 (Microsoft extended the length for ShellExecute in SP2).

  For example, the following command exceeds 128 characters (168 chars):

  ```
  C:\Program Files\Microsoft Office\OFFICE11\winword.exe c:\documents and settings\Greogory
  Appleblaught\My Documents\Draft 34 April 6 2006 Master License Agreement.doc
  ```

  The same path is much smaller using short path names (85 characters):

  ```
  C:\Progra~1\Micros~3\OFFICE11\winword.exe c:\docume~1\Greogo~1\MyDocu~1\Draft3~1.doc
  ```

The use of short path names in the registry is very common. For example, Microsoft Office installs hundreds of short path name references in the registry.

Short path names are not consistent across PCs and are also highly dependent on installation order. For example, installing Microsoft Office after installing Microsoft Visual Studio can create identical short path names that refer to different locations.

Installing Microsoft Office followed by Microsoft Visual Studio generates:

```
c:\progra~1\micros~1 and c:\progra~1\micros~2

c:\progra~1\micros~1 = c:\Program Files\Microsoft Office
c:\progra~1\micros~2 = c:\Program Files\Microsoft Visual Studio
```

When capturing a snapshot of an application's installation, VMware ThinApp writes a number of short path name values to the registry as described above. Moving the application to a different PC or installing additional applications on the same PC might affect the short path name values that the underlying operating system provides.

Virtualization solutions that are based on filter drivers do not have the ability to control short path name values because they are generated by the Windows file system. Because of this, a capture on one PC has a good chance of failing when moved to another PC or when executed later on the same PC after other applications have been installed.

Common areas of failure that occur when proper short path name support is not available include:

- **In-process and out-of-process COM failure** – The application tries to create COM objects and fails because the registry values point to the in-process or out-of-process COM server no longer point to the correct location.
- **An MSI is executed to reinstall an application even though it was fully installed** – `msi.dll` performs an integrity check for all MSI installed components requested by an application. If `msi.dll` detects that a DLL or data file is not located at the same location that it was originally installed to, it kicks off a reinstallation procedure to try to correct the problem.
- **Child processes fail to execute** – Many child process applications are executed using registry values that point to short path names. If the values do not point to the correct location, the child process fails to launch. This often renders an application unusable.
- **An application fails to load** – Applications might fail or do not load properly because they cannot locate the installation path or shared DLL paths using registry data.
- **An application indicates there are missing data files** – An application might display error messages relating to missing files because it can no longer locate a file using a short path name pointer from the registry.

Solving the Short Path Name Migration Problem

ThinApp uses a combination of dynamic registry data expansion and virtual short path names to enable applications to instantly migrate from one PC to another.

When an application writes a value to the registry, ThinApp scans the data for references to a short path name or shell folder location. If the application does this, VMware ThinApp stores the value internally in macro format that encodes both the shell folder location as well as the long path name value for a short path name.

At runtime, when an application queries the same registry value, it receives the original value it wrote to the registry. If the application moves to a different PC where the short path names are different, it obtains an automatically adjusted value.

For example, if you look at `HKEY_LOCAL_MACHINE.txt` for a capture of Microsoft Office, you see some entries similar to the following:

```
isolation_full HKEY_LOCAL_MACHINE\Software\Classes\CLSID\{03B54468-0899-4233-
8689-623FFFC295EE}\InprocServer32
Value= REG_SZ~%ProgramFilesDir~0032\Common Files\Microsoft Shared\Smart Tag\IETAG.DLL#2300
Value=ThreadingModel
REG_SZ~Apartment#2300
```

In this case, the registry value is encoded to expand to the shell folder location `c:\Program Files` and `\common files\microsoft shared\smart tag\ietag.dll` as in the following expression:

```
GetShortPathName(ExpandMacro("%ProgramFilesDir%") + "\Common Files\microsoft
shared\smart tag\ietag.dll")
```

Because ThinApp performs this macro expansion and collapse for every registry operation, it has been highly optimized to require very little CPU overhead. The expansion and collapse occurs dynamically at runtime, so no startup overhead time for applications with large registries occurs.

Preventing Conflicts When Isolation Is Used

In a virtual environment where an application is isolated from a PC, the normal file system is not aware of short path names that a virtualized application needs to use (for example, when a desktop PC has Microsoft Office installed natively). In this case, Microsoft Office occupies the short path name `c:\progra~1\micros~1`.

Suppose that, on this same desktop, a virtual version of Microsoft Visual Studio application begins running. In this case, Visual Studio needs to use a different short path name other than `c:\progra~1\micros~1` because this name is already in use.

The application must reserve the short path name `c:\progra~1\micros~2` when the short path name is allocated by the Windows file system. A problem arises if a third Microsoft application is installed while Visual Studio is running. It is uncertain if the process will receive a micros~3 or micros~2 short path name. In the latter case, it conflicts with the virtual application and the virtual application might fail.

ThinApp does not use filter drivers and does not depend on the Windows file system for the allocation of short path names. ThinApp prevents conflicts between short path names between virtual applications and system applications by using its own short namespace that does not collide with system applications.

Because each virtual application is isolated from all other virtual applications, you can have two virtual applications using the same short path names internally. ThinApp uses a different namespace than Windows for short path names when the system does not have the paths already created. You can see this in action by running a packaged `cmd.exe` and use `dir /x` to list short path names.

When you use ThinApp Setup Capture to capture applications, you can install captured applications by launching them from their default location without concern for short path names.

Virtual Services

ThinApp can be used to package services in a virtual service or a real Windows service. Virtual services run per user and real services run per machine.

Using Virtual Services

Virtual services have the following features:

- They are packaged together with a main supporting application.
- They run on application start-up.
- They run under the user account that runs the application.
- They can be started and stopped only by virtual applications in the same sandbox (visible to the services control panel services manager when you run from virtual `cmd.exe).`
- They do not require installation- or system-related changes.

By default, any application that installs a service automatically uses virtual services. When you use Setup Capture to capture an application that installs virtual services, no changes are needed to your project to run virtual services.

Starting Virtual Services

ThinApp uses the registry values captured by Setup Capture to determine if it should automatically start a virtual service when its host application is executed.

If the Service Startup Type is Automatic, ThinApp starts the service automatically before executing the host application.

If the Service Startup Type is Manual, ThinApp does not start the service until a host application specifically starts the service using service API calls (`OpenService` or `StartService`). To prevent a virtual service from being automatically started when the first parent process is launched, you can use the `AutoStartServices` option.

Stopping Virtual Services

Virtual services that are started during application startup are automatically shut down when the last non-service application exits. You can force virtual services to continue running until the user logs off using the AutoShutdownServices option.

Virtual services will continue running until you log out or when an application explicitly tells a service to stop.

Using a Real Windows Service

Real Windows services have the following features:

- They are packaged as a separate captured executable files.
- They run on boot-up.
- They run under the account specified by the service.
- They are directly visible to the control panel services manager.
- They can be started and stopped by any application.
- They require global system registry changes.